Reber further discloses that in convential systems, when source material is captured, information about the clip (i.e., "media data") and its actual digitized data are either coresident or linked directly together (Col. 2, lines 19-24). When the clip is manipulated or edited, the media data that is tied to it is used (Col. 2, lines 29-31). Any information about the source from which the particular clip came would also need to be stored with each clip (Col. 2, lines 31-33). As a result, all clips would be needed at any time to determine the breadth of any source relationships and, if any new source relationships were developed, it would be difficult to inform all the clips of the new information (Col. 2, lines 33-37). Additionally, it would be necessary to duplicate media data if certain clips or segments overlapped or were contained entirely within one another (Col. 2, lines 37-41).

To solve these problems, Reber discloses the system shown in Figure 1, which includes a media file manager (MFM) and a source manager (SM) (Col. 3, lines 51-64). A user accesses and operates on digitized media files by calls placed from the editing system to the MFM which creates and manages the media files (Col. 3, lines 64-66). The MFM interacts with the SM which maintains a table of relations between the linear media data recorded on source tapes and the digitized media files (Col. 3, line 66 – Col. 4, line 2). The MFM is responsible for management of all media that is present at any time and available to the system for use (Col. 4, lines 30-32). The MFM's purpose is to locate media specified by a user request. The user's request consists of a range of time from some specific source.

The MFM has an internal abbreviation of all media that is known to be accessible to it, and where and how that material may be retrieved (Col. 4, lines 46-49). This internal abbreviation may be a linked list which is initialized when the application is invoked (Col. 5, lines 20-24). The MFM initializes the linked list by scanning all disk drives on the system and determining if the media file database is valid by comparing the time stamp on each media file with the time stamp on the directory in which it was stored (Col. 5, lines 28-30). If the time stamps are equal, then the database is valid (Col. 5, lines 31-33).

A user request to the MFM uses the Mfm_handle call and is comprised of a source unique identifier or "id", a range on the source, type of media, and the physical channel requested if any that the source media was recorded from (Col. 5, lines 44-52). To process this

request, the MFM sifts through the linked list, and, if a match if a found then the handle to that record is returned to the requestor (Col. 5, lines 52-57).

Once the user has a handle to the requested record the user can use the call Mfm_open to actually obtain media data from the media source (Col. 6, lines 2-3). The user can also use the call Mfm_read to pass actual media data from the media source into a buffer specified by the caller (Col. 6, lines 10-12). The Mfm_close call is used to allow MFM to close the channel to the media source. If the media channel is open for write, the call examines a parameter which indicates caller specific information (Col. 6, lines 28-30).

B. Discussion of Kionka

Kionka is directed to a method and apparatus for optimizing computer file compilation. Kionka discloses that the aspect of the compilation process that determines which files to process and their order is called dependency analysis (Col. 1, lines 31-33). Dependency analysis determines the relationship between two independent objects (Col. 1, lines 33-34). An object depends on another object where a change in the latter requires a change in the former in order to bring each object into the current state (Col. 1, lines 35-37).

Kionka discloses a tree abstraction apparatus and method for determining the dependencies of source code files residing in multiple directories and for determining the optimal compilation order (Col. 2, line 15-19). The tree abstraction method extracts from a series of incomplete local hierarchical directories the critical inputs and intermediate inputs for a given set of main outputs from each directory, and creates a minimal description of the global hierarchy that expresses the logical dependent relations between the directories (Col. 2, lines 17-23). This global description is a minimal description because it expresses only the logical inter-directory dependency relations; it is optimized because it establishes the order in which the objects in various directories must be processed in order to update the entire system in a single pass, without repeated or unnecessary processing (Col. 2, lines 23-29).

C. The Combination of Reber and Kionka is Improper

The combination of Reber and Kionka is improper because the Office Action fails to establish a *prima facie* case of obviousness. MPEP §2142 requires that to establish a *prima facie*

case of obviousness, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings. The teaching or suggestion to make the claimed combination must be found in the prior art, and not based on applicant's disclosure. Further, when the motivation to combine the teachings of the references is not immediately apparent, it is the duty of the examiner to properly explain why the combination of the teachings is proper. MPEP §2142, pg. 2100-124, 8th Edition, Rev. 1, Feb. 2003.

The Office Action asserts that it would have been obvious to one of ordinary skill in the art at the time the invention was made to modify the teaching of Kionka into Reber, "because identifying the dependent files would allow the user to compare all the files that are related in order to determine the latest version to provide it to the user requests." However, the Office Action has failed to identify any teaching or suggestion in Reber that would motivate one to combine the system and method of maintaining a dynamic link between a media clip and its source file, as disclosed by Reber, with the system and method of determining dependencies between computer source code files for the purpose of determining an appropriate order of compilation, as disclosed by Kionka.

Additionally, the Office Action's assertion that "identifying the dependent files would allow the user to compare all the files that are related in order to determine the latest version to provide to the user requests" is inaccurate. Reber does not disclose or suggest that dependencies exist between files. The Office Action appears to confuse a dependency between files with an association between a media clip and media file. Reber discloses creating a "handle" for a media clip that is associated with a particular portion of the source media to link the media clip and the media file. This association between the media clip and the media file is very different from a dependency between two or more files. That is, in Reber there is no disclosure or suggestion that the data of one media file is dependent on the data of another media file. Reber merely discloses that a handle is created which is used to refer to a particular media file. Thus, there is an association between the handle and the file. This is very different from a dependency between files. Therefore, there is no teaching or suggestion in Reber that would have motivated one of ordinary skill in the art to combine Kionka's method of analyzing dependencies between

computer source code files with Reber's system of creating and maintaining an association between a media clip and a media file.

The Office Action further asserts, as motivation for combining the two references, that using the dependency analysis method of Kionka in the system of Reber would aid in the comparing of files. However, Reber does not disclose or suggest comparing files and, indeed, comparing media files would provide no improvement in the system of Reber. The Office Action appears to refer to the Mfm_init call discussed at Col. 5, lines 21-42 of Reber as basis for the assertion that Reber "compares files" to "determine the latest version to provide it to the user requests." However, this paragraph simply discloses how the linked list which the MFM uses to locate user requested media clips is initially created. The MFM initializes the linked list by scanning all disk drives on the system and determining if the media file database is valid by comparing the time stamp on each media file with the time stamp on the directory in which it was stored (Col. 5, lines 28-30). Thus, the MFM does not compare files. Instead, the MFM compares a time stamp on a file with the time stamp on the directory in which the file is contained. Because in Reber there is no dependency between media files, there is no reason to compare media files to determine dependencies. Because there is no disclosure or suggestion of file dependencies in Reber, one of ordinary skill in the art would not have been motivated to use a file dependency analysis technique, as disclosed by Kionka, as this technique would serve no purpose in the system of Reber. Indeed, Applicant is at a loss to even speculate what advantage a comparison of such files would have in the system of Reber.

Thus, because the Office Action fails to establish a *prima facie* case of obviousness for the combination of Reber and Kionka, it is respectfully requested that the rejections of independent claims 1 and 10 under 35 U.S.C. §103(a) be withdrawn. As claims 2-9 and 17-18 depend from claim 1 and claims 11-16 depend from claim 10, it is respectfully requested that the rejection of these claims also be withdrawn.

Should the Examiner maintain that the dependency analysis technique of Kionka may be used to determine if the shorthand version of Reber's media files are valid, then the Examiner is respectfully requested to point out, with particularity, the portion of Reber that is asserted to disclose that media files are dependent on one another. Further, as required by MPEP §2142, the Examiner is also respectfully requested to explain how a dependency analysis of the allegedly

dependent media files would aid one in determining whether the shorthand version of a media file is valid and explain how one would incorporate the tree abstraction technique for dependency analysis disclosed by Kionka into the system of Reber to accomplish this purpose.

D. Even If Reber and Kionka are Combined, Claims 1 and 10 Patentably Distinguish Over The Combination

Even assuming, *arguendo*, that the combination of Reber and Kionka is proper, the references, whether taken alone or in combination, fail to disclose every limitation of claims 1 and 10.

1. Discussion of Claim 1

Claim 1 is directed to a method of operating a computer system to validate the data stored in a plurality of data files in a database, each of said data files having an associated file type and being arranged in a plurality of data stores in said data base. At least one of said data files is a data dependent file containing data dependent on data in one or more other files of said data store. The method of claim 1 comprises steps of selecting a file locator which is associated with a respective one data store in said data base; via said selected file locator identifying a first dependent file and identifying one or more other files on which said first file is dependent; for each identified file, selecting a first file reader associated with the file type of the identified file; via each said selected first file reader, determining a predetermined parameter of said identified file; comparing the predetermined parameter from the first file with that from the or each other file; and responsive to said comparison step, providing an output signal for each data file indicating whether the data file is valid.

The Office Action asserts that Reber discloses each limitation of claim 1, except that Reber does "not explicitly teach dependent data." The Office Action further asserts that Kionka teaches dependent data. Preliminarily, Applicant notes that the rejection of the Office Action is inconsistent. The Office Action initially asserts that Reber discloses at Col. 5, lines 52-56, the limitation of claim 1 that recites, "via said selected file locator identifying a first dependent file and identifying one or more other files on which said first file is dependent." However, the Office Action later asserts that "Reber does not explicitly teach dependent data." If Reber does not teach dependent data, then Reber cannot disclose "identifying one or more file on which said

first file is dependent." Because Reber fails to disclose or suggest this limitation and Kionka fails to cure this infirmity of Reber, claim 1 patentably distinguishes over the combination of Reber and Kionka.

The Office Action further asserts that the limitation of claim 1 that recites "via each said selected first file reader, determining a predetermined parameter of said identified file" is disclosed by Reber at Col. 6, lines 28-32. As discussed above in relation to the summary of the Reber reference, Col. 6, lines 28-32 of Reber discloses the Mfm_close function call. This function call is used to close a channel between the caller and the MFM. If the channel is open for write (as opposed to open for read) then the call examines a parameter which indicates caller specific information. As is well known in the art, function calls often include parameters. As used in this context, parameters are data provided by the caller of the function, to be used by the function. For example, suppose that a function called "sum" returns to the caller the sum of any two operands specified by the caller. The caller would specify these two operands as parameters to the function. Thus, the parameter referred to in Col. 6, line 29 is a parameter to the Mfm_close function, not a parameter of the media file associated with the MFM_GIST handle that is passed to the function, as the Office Action appears to suggest.

By contrast, claim 1 recites, "via each said selected first file reader, determining a predetermined parameter of said identified file." A parameter of a file is very different than a parameter to a function call. Reber fails to disclose determining a predetermined parameter of the identified file. Kionka fails to cure this infirmity of Reber, as Kionka fails to disclose or suggest using a file reader to determine a predetermined parameter of the identified file.

The Office Action next asserts that the limitation of claim 1 that recites, "comparing the predetermined parameter from the first file with that from the or each other file," is disclosed by Reber at Col. 5, lines 20-35. As discussed above, this cited paragraph of Reber discloses the Mfm_init function call. Reber discloses that this function compares the time stamp on the file with the time stamp on the directory. Thus, Reber discloses comparing an attribute of a file with an attribute of a directory. Further, this attribute (i.e., the time stamp) is unrelated to the parameter of Mfm_close function call which identifies caller specific information.

By contrast, this limitation claim 1 is directed to comparing parameters between two or more files, not between a file and a directory. The parameter of the first file that is compared is

the same parameter that was determined using the first selected file reader. However, the parameter in Reber which indicates caller specific information is not the same as the time stamp of the file which is compared to the time stamp of the directory in the Mfm_init function of Reber. Kionka does not cure this infirmity of Reber, as Kionka fails to disclose or suggest, "comparing the predetermined parameter from the first file with that from the or each other file."

Further, Reber discloses comparing the time stamp of the file to the time stamp of the directory for the purpose of determining whether the shorthand abbreviation of the database is valid. If the shorthand abbreviation is invalid, than Reber discloses simply recreating it.
By contrast, claim 1 recites, "providing an output signal for each data file indicating whether the data file is valid." However, Reber does do not disclose indicating whether a media file is valid, Reber simply discloses indicating whether the abbreviation of the database is valid, regardless of the status of the data in the media files.

For at least these reasons, claim 1 patentably distinguishes over the Reber and Kionka references, whether taken alone or in combination. Accordingly, it is respectfully requested that the rejection of claim 1 under 35 U.S.C. §103(a) be withdrawn.

Claims 2-9 and 17-18 depend from claim 1 and are patentable for at least the reasons discussed above in connection with claim 1. Accordingly, it is respectfully requested that the rejections of claims 2-9 and 17-18 be withdrawn.

2. Discussion of Claim 10

Claim 10 is directed to a computer system arranged to validate data stored in a plurality of data files in a data base, each of said data files having an associated file type and being arranged in a plurality of data stores in said data base. At least one of said data files is a data dependent file containing data dependent on data in one or more other files of said data base. The system comprises a plurality of file locators, each associated with a respective data store in said data base and arranged to identify a first data dependent file in said associated data store and one or more other files in said data base on which said first file is dependent. The computer system further comprises a plurality of file readers, each associated with a respective file type and each arranged to determine a predetermined parameter for at least one identified file having that associated file type; comparison means arranged to compare the predetermined parameter

determined for said first file with the predetermined parameter determined for each other file; and output means, responsive to said comparison means and having an output which indicates whether said first file is valid.

Similar to the rejection of claim 1, the rejection of claim 10 is also inconsistent. The Office Action asserts that Reber discloses each limitation of claim 10, except that Reber does "not explicitly teach dependent data." The Office Action initially asserts that Reber discloses at Col. 5, lines 52-56, the limitation of claim 10 that recites, "a plurality of file locators each associated with a respective data store in said database and arranged to identify a first data dependent file in said associated data store and one or more other files in said database on which said first file is dependent." However, the Office Action later asserts that "Reber does not explicitly teach dependent data." If Reber does not teach dependent data, then Reber cannot disclose a plurality of file locators arranged to identify "one or more file on which said first file is dependent." Because Reber fails to disclose or suggest this limitation and Kionka fails to cure this infirmity of Reber, claim 10 patentably distinguishes over the combination of Reber and Kionka.

The Office Action further asserts that the limitation of claim 10 that recites "a plurality of file readers each associated with a respective file type and each arranged to determine a predetermined parameter for at least one identified file," is disclosed by Reber at Col. 6, lines 28-32. However, as discussed above, the parameter referred to in Col. 6, line 29 is a parameter to the Mfm_close function. The parameter is not a parameter of the media file associated with the MFM_GIST handle that is passed to the function, as the Office Action appears to suggest.

A parameter of a file is very different than a parameter to a function call. Reber fails to disclose file readers "arranged to determine a predetermined parameter for at least one identified file." Kionka fails to cure this infirmity of Reber, as Kionka fails to disclose or suggest using a file reader to determine a predetermined parameter of said identified file.

The Office Action next asserts that the limitation of claim 10 that recites, "comparison means arranged to compare the predetermined parameter determined for said first file, with the predetermined parameter determined for each other file" is disclosed by Reber at Col. 5, lines 20-35. As discussed above, this paragraph of Reber discloses the Mfm_init function call. Reber discloses that this function compares the time stamp on the file with the time stamp on the

directory. Thus, Reber discloses comparing an attribute of a file with an attribute of a directory. Further, this attribute (i.e., the time stamp) is unrelated to the parameter of Mfm_close function call which identifies caller specific information.

By contrast, this limitation claim 10 is directed to comparing parameters between two or more files, not between a file and a directory. The parameter of the first file that is compared is the same parameter that was determined using the first selected file reader. However, the parameter in Reber which indicates caller specific information is not the same as the time stamp of the file which is compared to the time stamp of the directory in the Mfm_init function of Reber. Kionka does not cure this infirmity of Reber, as Kionka fails to disclose or suggest, "comparison means arranged to compare the predetermined parameter determined for said first file, with the predetermined parameter determined for each other file."

Additionally, as discussed above in connection with claim 1, Reber discloses comparing the time stamp of the file to the time stamp of the directory for the purpose of determining whether the shorthand abbreviation of the database is valid. If the shorthand abbreviation is invalid, than Reber discloses simply recreating it. By contrast, claim 10 recites, "output means responsive to said comparison means and having an output which indicates whether said first file is valid." However, Reber does do not disclose indicating whether a media file is valid, Reber simply discloses indicating whether the abbreviation of the database is valid, regardless of the status of the data in the media files.

For at least these reasons, claim 10 patentably distinguishes over the Reber and Kionka references, whether taken alone or in combination. Accordingly, it is respectfully requested that the rejection of claim 10 under 35 U.S.C. §103(a) be withdrawn.

Claims 11-16 depend from claim 1 and are patentable for at least the reasons discussed above in connection with claim 1. Accordingly, it is respectfully requested that the rejections of claims 11-16 be withdrawn.

CONCLUSION

In view of the foregoing amendments and remarks, this application should now be in condition for allowance. A notice to this effect is respectfully requested. If the Examiner

believes, after this amendment, that the application is not in condition for allowance, the Examiner is requested to call the Applicant's attorney at the telephone number listed below.

If this response is not considered timely filed and if a request for an extension of time is otherwise absent, Applicant hereby requests any necessary extension of time. If there is a fee occasioned by this response, including an extension fee, that is not covered by an enclosed check, please charge any deficiency to Deposit Account No. 23/2825.

Respectfully submitted,
David SMITH, Applicant

By: _____

Robert A. Skrivanek, Jr., Reg. No. 41,316
Wolf, Greenfield & Sacks, P.C.
600 Atlantic Avenue
Boston, Massachusetts 02210-2211
Telephone: (617) 720-3500

Docket No. S01022.80602.US
Date: April 21, 2003
x04/21/03 (04/20/03 having fallen on a Sunday)

MARKED-UP SPECIFICATION

The paragraph beginning at page 11, line 11 has been amended as follows:

dependency_record = main_file_info ":" sub_file_info ["," sub_file_info] ";"
main_file_info = main[_]file[_]filenamemain_file[_]filetype [(main_file_other_param)]
sub_file_info = sub_file_filename sub_file_filetype [(sub_file_other_param)]

The paragraph beginning at page 13, line 9 has been amended as follows:

generate_directory_depfile ("$library_root/$module_name",
"epic_netlist", "\\.epic\[S]$");

The paragraph beginning at page 18, line 33, and ending on page 19, line 2 has been amended as follows:

Next the core system [in] invokes the generator for the particular library type 303. The library generator then calls the core system (via generator Utils) to extract dependency information from the synthesis log, RTL results and netlist results. The core system then calls the generator_depfile routines 304 from the relevant parsers for each of the files thereby generating the dependency files.

MARKED-UP CLAIMS

10. (Amended) A computer system arranged to validate data stored in a plurality of data files in a data base each of said data files having an associated file type and being arranged in a plurality of data stores in said data base, wherein at least one of said data files is a data dependent file containing data dependent on data in one or more other files of said data base, said system comprising:

a plurality of file locators each associated with a respective data store in said data base and arranged to identify a first data dependent file in said associated data store and one or more other files in said data base on which said first file is dependent;

a plurality of file readers each associated with a respective file type and each arranged to determine a predetermined parameter for at least one identified file having that associated file type;

comparison means arranged to compare the predetermined parameter determined for said first file[,] with the predetermined parameter determined for each other file; and

output means responsive to said comparison means and having an output which indicates whether said first file is valid.